International Journal of Engineering Research and Development
***e*-ISSN: 2278-067X, *p*-ISSN: 2278-800X, www.ijerd.com**
Volume 4, Issue 2 (October 2012), PP. 01-09

Fuzzy-based approach in Clustering Data Sources- a Case Study

V V Siva Rama Raju[1], P B Siva Varma[2], K R S Ramaraju[3], T V K P Prasad[4]
[1,2,3,4] *Assistant Professor, Department of CSE, S.R.K.R Engg. College, Affiliated to Andhra University, Bhimavaram, W.G.District, Pin-534 204, A.P. INDIA.*

Abstract:-In order to improve the effectiveness of community-wide cooperation, the capability of extracting sharable knowledge from heterogeneous documents produced by multiple sources and in a timely fashion is therefore crucial. To this kind of demand is to supply a suite of tools for the end user and for the developer as well, to create, maintain and navigate an ontology generated from an varied amount of data. To wrap the set of data sources in a structured format, such as XML and, using fuzzy techniques in order to compare the incoming dataflow, cluster them in a semi-automatic way. The final purpose is to generate a browsable ontology of the processed resources. This paper describes a theoretical approach on data mining, information classifying and present a way of building Offline Browsing on fuzzy based clustering and ontology's, defining concepts as clusters of concrete XML objects based on fuzzy approach.

Keywords:-Clustering, Fuzzy approach, Offline Browsing, XML;

I. INTRODUCTION

The need to have an easy access and a reliable classification of the amount of data available nowadays on the web is one of the challenges of the future directions of data mining. The community of users has to deal with a huge amount of digital information such as text or semi-structured documents in the form of web pages, reports, papers and e-mails. In order to improve the effectiveness of community-wide cooperation, the capability of extracting sharable knowledge from heterogeneous documents produced by multiple sources and in a timely fashion is therefore crucial. To this kind of demand is to supply a suite of tools for the end user and for the developer as well, to create, maintain and navigate an ontology generated from an varied amount of data. To wrap the set of data sources in a structured format, such as XML and, using fuzzy techniques in order to compare the incoming dataflow, cluster them in a semi-automatic way. The final purpose is to generate a browsable ontology of the processed resources. In this section describing our tool will be described the implementation of the fuzzy elements and algorithms are used, for who those wants to use them for his own fuzzy-based applications[1,2].

A. History

Fuzzy Logic (FL) is was conceived by Lotfi Zadeh, a professor at the University of California at Berkley, and presented not as a control methodology, but as a way of processing data by allowing partial set membership rather than crisp set membership or non-membership. This approach to set theory was not applied to control systems until the 70's due to insufficient small-computer capability prior to that time. Professor Zadeh reasoned that people do not require precise, numerical information input, and yet they are capable of highly adaptive control. If feedback controllers could be programmed to accept noisy, imprecise input, they would be much more effective and perhaps easier to implement. Unfortunately, U.S. manufacturers have not been so quick to embrace this technology while the Europeans and Japanese have been aggressively building real products around it [3,4].

In this context, FL is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. It can be implemented in hardware, software, or a combination of both. FL provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. FL's approach to control problems mimics how a person would make decisions, only much faster [5].

i. Working of Fuzzy Logic

FL requires some numerical parameters in order to operate such as what is considered significant error and significant rate-of-change-of-error, but exact values of these numbers are usually not critical unless very responsive performance is required in which case empirical tuning would determine them. For example, a simple temperature control system could use a single temperature feedback sensor whose data is subtracted from the command signal to compute "error" and then time-differentiated to yield the error slope or rate-of-change-of-error, hereafter called "error-dot". Error might have units of degs F and a small error considered to be 2F while a large error is 5F. The "error-dot" might then have units of degs/min with a small error-dot being 5F/min and a large one being 15F/min. These values don't have to be symmetrical and can be "tweaked" once the system is operating in order to optimize performance. Generally, FL is so forgiving that the system will probably work the first time without any tweaking.

FL was conceived as a better method for sorting and handling data but has proven to be a excellent choice for many control system applications since it mimics human control logic. It can be built into anything from small, hand-held

products to large computerized process control systems. It uses an imprecise but very descriptive language to deal with input data more like a human operator. It is very robust and forgiving of operator and data input and often works when first implemented with little or no tuning[6,7].

B. Use of fuzzy logic

FL offers several unique features that make it a particularly good choice for many control problems.

- It is inherently robust since it does not require precise, noise-free inputs and can be programmed to fail safely if a feedback sensor quits or is destroyed. The output control is a smooth control function despite a wide range of input variations.
- Since the FL controller processes user-defined rules governing the target control system, it can be modified and tweaked easily to improve system performance.
- New sensors can easily be incorporated into the system simply by generating appropriate governing rules.
- FL is not limited to a few feedback inputs and one or two control outputs, nor is it necessary to measure or compute rate-of-change parameters in order for it to be implemented. Any sensor data that provides some indication of a system's actions and reactions is sufficient. This allows the sensors to be inexpensive and imprecise thus keeping the overall system cost and complexity low.
- Because of the rule-based operation, any reasonable number of inputs can be processed and numerous outputs generated, although defining the rulebase quickly becomes complex if too many inputs and outputs are chosen for a single implementation since rules defining their interrelations must also be defined. It would be better to break the control system into smaller chunks and use several smaller FL controllers distributed on the system, each with more limited responsibilities.
- FL can control nonlinear systems that would be difficult or impossible to model mathematically. This opens doors for control systems that would normally be deemed unfeasible for automation.
-

II. RELATED WORK

C. Clustering

Clustering analysis finds clusters of data objects that are similar in some sense to one another. The members of a cluster are more like each other than they are like members of other clusters. The goal of clustering analysis is to find high-quality clusters such that the inter-cluster similarity is low and the intra-cluster similarity is high.

Clustering, like classification, is used to segment the data. Unlike classification, clustering models segment data into groups that were not previously defined. Classification models segment data by assigning it to previously-defined classes, which are specified in a target. Clustering models do not use a target. Clustering is useful for exploring data. If there are many cases and no obvious groupings, clustering algorithms can be used to find natural groupings. Clustering can also serve as a useful data-preprocessing step to identify homogeneous groups on which to build supervised models. Clustering can also be used for anomaly detection. Once the data has been segmented into clusters, you might find that some cases do not fit well into any clusters. These cases are anomalies or outliers [8-10].

D. Clustering Algorithms

Clustering algorithms may be classified as listed below:

- Exclusive Clustering
- Overlapping Clustering
- Hierarchical Clustering
- Probabilistic Clustering

In the first case data are grouped in an exclusive way, so that if a certain datum belongs to a definite cluster then it could not be included in another cluster. A simple example of that is shown in the figure below, where the separation of points is achieved by a straight line on a bi-dimensional plane. On the contrary the second type, the overlapping clustering, uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership. In this case, data will be associated to an appropriate membership value. Instead, a hierarchical clustering algorithm is based on the union between the two nearest clusters. The beginning condition is realized by setting every datum as a cluster [11-13]. We study four of the most used clustering algorithms:

- K-means
- Fuzzy C-means
- Hierarchical clustering
- Mixture of Gaussians

Each of these algorithms belongs to one of the clustering types listed above. So that, K-means is an *exclusive clustering* algorithm, Fuzzy C-means is an *overlapping clustering* algorithm, Hierarchical clustering is obvious and lastly Mixture of Gaussian is a *probabilistic clustering* algorithm. We will discuss about each clustering method in the following paragraphs.

E. Distance Measure

An important component of a clustering algorithm is the distance measure between data points. If the components of the data instance vectors are all in the same physical units then it is possible that the simple Euclidean distance metric is

sufficient to successfully group similar data instances. However, even in this case the Euclidean distance can sometimes be misleading. Figure 1 shown illustrates this with an example of the width and height measurements of an object. Despite both measurements being taken in the same physical units, an informed decision has to be made as to the relative scaling. As the figure shows, different scaling can lead to different clustering.

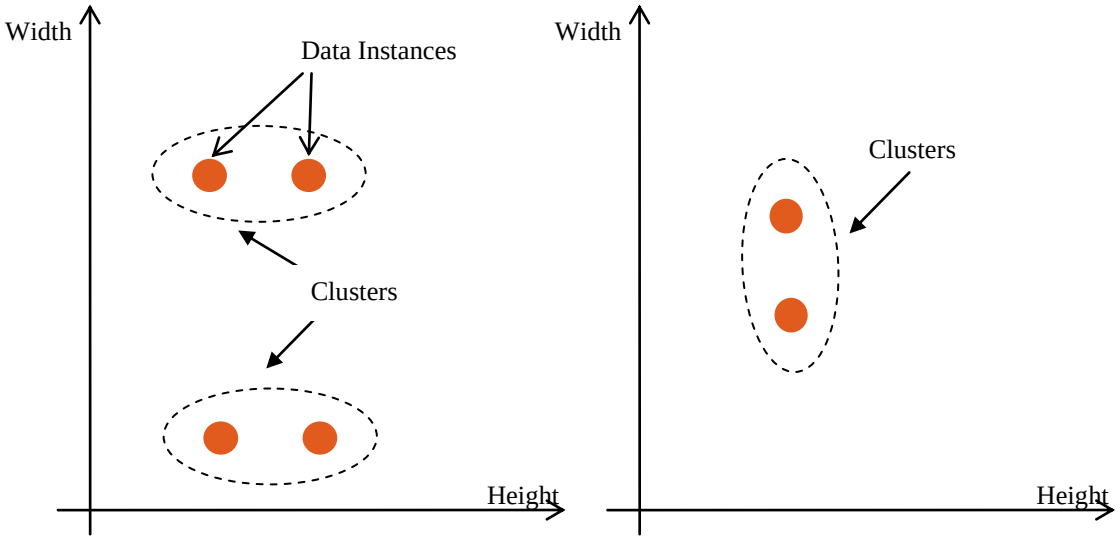

Fig 1 Different scaling ***a) Before Scaling*** ***B) After Scaling***

F. Analysis

Internet is global network of networks. It is essentially a communication tool that offers immediate access to people and information. Documents that are viewed on the net are mostly written in HTML (Hyper Text Markup Language) and are called web pages. This language is used to create hypertext documents that have hyper-links embedded in them. There are two important entities involved here are the clients and the server. The clients, in the simplest case retrieve data from the server and display it. Servers respond to the requests for data. Web Browsers retrieve data on demand. The user asks for a page at a URL (Uniform Resource Locator) and the browser gets it. Search programs that run on a single client system are called spiders. A spider downloads a page at a particular URL, extracts the URLs from the links on that page, downloads the pages referred many things based on the above principle like indexing the URLs in a database or hunting for specific information. The main purpose is to generate a browsable ontology of the processed resources. The project is developed with the main intention to provide the user with a tool, which provides them an integrated set of services when dealing with a particular Web site[14,15].

G. User-Domain

Offline Browsing: A Fuzzy-Based Approach in Clustering Data Sources and Ontology Metadata can be useful to lot many people. Some of them include

Students: A specific website containing the information related to the students work can be downloaded instead of saving the contents of each and every page. It is especially useful while gathering information for projects, preparing for seminars, examinations.

Software developers: It helps in bulk download of Tutorials for the Newest Technologies.

Network Administrators: It reduces the Bandwidth Cost by letting the frequently visited Web–sites to be shared over the Intra-net.

III. OBJECTIVES

Improving the reliability of downloading the website even in cases of the Net or the server going down between the download processes. Offline Browser is a designed to run under any Operating System as it is developed using the JAVA language. The platform independent byte code of JAVA lets the application to run on any platform with minimal or no change.

- The primary objective of the project is to provide reliable classification of the amount of data available on the web.
- Improving the reliability of downloading the web site even in cases of the Net or the server going down between the download processes.

There are already some applications which do the same tasks i.e. help in offline browsing. For example Meta-products Offline Browser , this formed a part of our study for developing the present application. We have used the application and were able to observe the drawbacks of this system. The study of this product has helped us in designing our application to overcome some of the drawbacks. The drawbacks that are faced by existing systems include

- Most of the Applications are targeted for a specific platform like the windows. So if a user is switching between the platforms i.e. the operating systems, he requires two different versions of the same product developed for the different platforms the user is working on.

- Better page management can be done in cases when the Net connection breaks down in between the download process.
- These applications are commercial products, which mean that it is not easy for students to access them. Even if they can get a free downloaded version they have access to a limited number of features.

In this paper, we propose a system, which overcomes some of the drawbacks

- The application is being developed in JAVA language, which helps in generation if platform independent Byte Code. Thus the application would be portable to any environment where a Java Virtual Machine (JVM) exists.
- We have tried to manage the pages better by introducing a project log file which helps in download of the website in cases when the net connection breaks down. It follows a protocol by which the incomplete downloaded websites are downloaded again when the Net connection is established without the users no.

The aim of the present paper is to build Offline Browsing on fuzzy based clustering and ontologies, defining concepts as clusters of concrete XML objects. Our rough Fuzzy based clustering on simple structured based classification and content based approaches. It lets us work at high level abstraction and offers a means for dealing with imprecise measurement of data. we show how Offline Browsers retrieve data on demand. The user asks for a page at a URL (Uniform Resource Locator) and the browser gets it Search programs that run on system downloads a page at a particular URL, extracts the URLs from the links on that page, downloads the pages referred many things based on the above principle like indexing the URLs in a database on our Fuzzy-Based Approach in Clustering with inexact facts. We are now ready to organize extracted information in a standard knowledge representation format with ontology metadata

IV. SYSTEM ARCHITECTURE

The project is developed with the main intention to provide the user with a tool, which provides them an integrated set of services when dealing with a particular Web site[16]. As illustrated earlier it has two main sections. Graphical User Interface for interacting with the user Backend portion: responsible for processing the user requests. It is shown in Fig 2.

H. Overview of Design

The Backend forms the core of the tool responsible for interacting with the outside environment i.e. the Internet. It mainly consists of the following parts

- Server, which acts as an integrator with the front end (Java6.0).
- Client, which is responsible for downloading the URLs from the Internet
- Parser, which searches the HTML pages and extracts the URLs.

Apart from the above parts we also have other utilities, which are a Queue, which is useful for

- Exchanging messages between different parts of the program and the server.
- Storing the extracted URLs form the HTML page in the download item queue for retrieval.

A Utility class which has various static functions that range from date conversion routines to determining whether a URL is absolute or not.

I. Design of the Server

The server forms the link between the User interface and the Service-rendering portion of the system. At any time the server is in a particular state, which depicts the state of the system. The presence of this state determines the way in which it processes its next message.

To make the application less complex we have designed the server as a state machine which can process or carry on only one instance of the service at any given time i.e. although it is principally possible to carry out the Web-Mirror service on different URLs simultaneously, we have restricted that number to one.

This means that at any given time only one instance of the web mirror service would be running, which is passed the starting URL.It, implements the message constants interface, which provides a consistent coding of the message types into integer values. It extends the Thread class, as it has to run in a separate thread. It uses the message queue instance created at the start of the application. It waits for messages on this message queue and whenever it is notified about the insertion of a message it extracts the message object, determines the type of service required and invokes the respective service desired by instantiating that specific service or by calling the required function of that service object.

J. Design of the Client

The client portion of the back end is responsible for downloading the specified URL.The web mirror service starts a number of client threads to download the URLs in parallel each establishing a separate connection with the server. The very essence of this service includes parallelism because the network input –output is very slow compared to the local IO and CPU processing speed, so it makes very much sense to start the download process of the next URL in queue if the present connection is waiting for data to come or download. Since this cannot happen in the same thread of execution so a separate thread is required.

Initially a thread pool is created which waits on the download item queue, which is initially empty. Then a download item object is created with the starting URL given by the server and is inserted into the download queue and the required objects waiting on this download item queue are notified about the insertion. One of the threads in the thread pool waiting on the queue retrieves the starting URL. If it is a HTML page then it passes the input-stream to the parser, other wise it downloads the URL file into the local secondary memory.

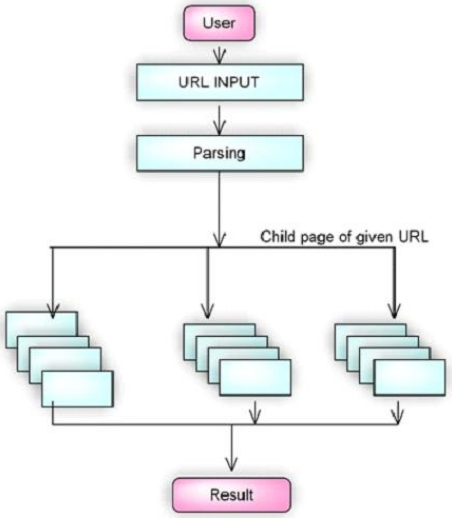

***Fig 2* Architecture flow of Offline Browsing**

K. Design of the Parser

The HTML parser is used for serving two main purposes. One is to extract the local URLs and convert the relative URLs into absolute URLs and insert them into the download item queue. The second purpose is to edit the URLs in the html pages being downloaded.

This is done in order to convert the absolute URLs present in the html pages into relative URLs so that the downloaded web site can be stored as a standalone website without requiring any outside support such as the Internet to browse through the information.

The parser also stores the content of the html page after editing the URLs into the output stream supplied by the user of the parser object.

V. IMPLEMENTATION

The following are the various modules:

- Fuzzy Representation of XML Data
- The Class Builder
- Structure-Based Document Classification
- Content-Based Document Classification

L. Fuzzy Representation of XML Data

The first step of our metadata extraction process is encoding XML documents (i.e., instances of our simplified Info set) as fuzzy multi sets, i.e., fuzzy bags. A bag is simply a collection of elements which may contain duplicates. As we shall see, bags provide a good flat encoding for XML documents since they can express the fact that the same tag can be repeated at different levels and/or with different cardinalities at the same level. Set-based flat encodings for XML documents were first proposed in our previous work on approximate XML querying as a way of avoiding carrying out computations on XML trees since tree matching problems have computational complexities ranging from polynomial to NP-complete. Intuition tells us that adopting a model based on bags rather than sets will make our technique more sensitive to differences in cardinality between tags of the same name. Again, intuitively, the similarity between bags will decrease when differences in elements' cardinalities increase, but the amount of this decrease may selectively depend on the specific tags involved, fine-tuning similarity values. Our encoding of an XML tree as a fuzzy bag is computed as follows: Supposing that every tag xi has an associated membership degree $\mu(x_i)$, initially equal to 1 ($\mu(x_i) = 1$), we divide every membership degree by its nesting level L (where Lroot = 1), obtaining a "lossy" representation of the XML tree that only takes into consideration the original nesting level of tags.4 Our flat encoding is also sensitive to structural differences between XML documents containing the same group of tags in different positions. Applying our encoding to the tree representation of the XML document A.xml, we obtain the fuzzy bag:

A={1/R, 0.5/a, 0.5/c, 0.5/d, 0.33/a, 0.33/a, 0.33/b, 0.33/c; 0.33/e}

We use the following representation to indicate the presence of content information connected to leaf nodes:

A= {1/R, 0.5/a, 0.5/c[data], 0.5/d,(0.33, 0.33)/a[data], 0.33/a, 0.33/b[data], 0.33/c[data]; 0.33/e[data]}

M. The Class Builder

Our approach to the architecture is a suite of software tools aimed at domain analysis experts and end users alike. In particular, our Class Builder clustering tool uses fuzzy techniques in order to compare items in an incoming XML data flow, clustering them in a semiautomatic way. As we shall see, our Class Builder can be coupled with an auxiliary layer, called onto Extractor, in order to build/update ontologies using Semantic-Web-style metadata formats. In the next sections, we will illustrate the ideas and the algorithms behind our tools

N. Structure-Based Document Classification

A major feature of our flat encoding technique is that, unlike XML trees, fuzzy bags lend themselves to fast and efficient clustering. When comparing fuzzy bags representing XML documents several measures of object similarity can be used, some of them aimed at specific domains. Choosing the "right" similarity usually depends on domain and even data-set-dependent criteria. Also, applying the extension principle, it is possible to extend to fuzzy bags most object comparison measures originally defined for fuzzy sets. When performing this extension, it is important to note that, while the (single) occurrence of an element x in a fuzzy set has a fuzzy membership value $0 < \mu \leq 1$, the (possibly multiple) occurrences of an element x in a fuzzy bag A constitute a fuzzy set whose fuzzy cardinality is not a single value, but a fuzzy number of occurrences denoted $\Omega_A(x)$. The cardinality $|\hat{A}|$ of an ordinary fuzzy set A is defined by

$$\forall n \in \mathbb{N}, \quad \mu_{|\hat{A}|}(n) = \sup\{\alpha : |A_\alpha| \geq n\}.$$

By the extension principle, the cardinality of a bag is the arithmetic sum of the occurrences of its elements, including those which appear multiple times.

The cardinality of a fuzzy bag A is defined as follows:

$$|\hat{A}| = \sum_{x \in X} \Omega_A(x).$$

For example, for elements a and b in the bag $A = \{\langle 1, 0.1, 0.1\rangle/a, \langle 0.5\rangle/b\}$, we get:

$$\begin{aligned} \Omega_A(a) &= \{1/0, 1/1, 0.1/2, 0.1/3\}, \\ \Omega_A(b) &= \{1/0, 0.5/1\}, \\ |\hat{A}| &= \Omega_A(a) + \Omega_A(b) \\ &= \{1/0, 1/1, 0.5/2, 0.1/3, 0.1/4\}. \end{aligned}$$

As far as structural similarity is concerned, we need one that is monotonic with respect to element addition and removal, such as the Jaccard norm:

$$S(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Bags are clustered using S as a similarity by means of a hybrid k-Means and NN clustering algorithm. Specifically, our Class Builder uses a tunable α threshold for the clustering process, in order to avoid suggesting an initial number of clusters (k). This way, some well-known problems of the k-Means algorithm are entirely avoided. Our clustering algorithm compares all items with the centroid of each cluster considering only the top similarity value. If this value is bigger than α, the data item is inserted in the cluster; otherwise, a new empty cluster is generated and the item is inserted into it. Class Builder offers two different ways to calculate the centroid of each cluster, called cluster head. The former method chooses the items represented by the smallest fuzzy bag so that a cluster centroid always corresponds to a real data item. The latter generates a new fuzzy bag as the union of all fuzzy bags in the cluster. This way, a cluster head does not necessarily coincide with a real data item since each tag is taken with the cardinality and membership degree it has in the bag where it appears with the highest cardinality.

O. Content-Based Document Classification

While traditional content-based clustering operates on document-centric data, our content-based clustering technique must work on a collection of semi structured tagged documents. So, we start by computing content-based similarity at the tag level, comparing contents of the same tag in two different documents. Content-based similarity at the document level can then be obtained by aggregating tag level similarity values. Since XML documents are encoded as fuzzy bags, our comparison works on content data attached to bag elements, whose original nesting level (in the XML tree structure) is encoded by the membership degree. The following figure shows the tree representations of two structurally identical fuzzy sets A and B. Since we are evaluating content similarity, we limit our interest to subsets containing leaf nodes only (i.e., nodes where content information can be found). With {x[data]}D, we designate content data delimited by tag x in document D. In order to evaluate content similarity, we combine two different functions. The first one is a tag-level similarity f comparing data belonging to tags with the same name in different documents: f_x ({x[data]}D1 , {x[data]}D2).

Since terms may have different informative value depending on the tag they belong to, we also take into account the membership value μ (xi) associated to the i^{th} tag. When dealing with text data, we represent the content of each tag by the well-known Vector Space Model (VSM), which represents natural language documents as vectors in a multidimensional space: Every document d is considered to be a vector d in the term space (set of document words). Each document is represented by the (TF) vector $d_{ft} = (f_{t1}, f_{t2}, \ldots, f_{tn})$, where f_{ti} is the frequency of the ith term in the document.

Then, we can use a standard cosine distance—expressed by the equation $\cos(d1, d2) = (d_1 \cdot d_2)/\|d_1\| \cdot \|d_2\|$, where '**.**' denotes the scalar product and $\|d\|$ is the Euclidean Length. However, text comparison hardly captures the real similarity between typed values such as dates and numerical types.

Our XML content can belong to one of the basic XML Schema data-types (URLs, numbers, dates, etc.) or be a text blob. In the first case, it is useful to apply type-driven matching, i.e., compute data similarity by taking into account their estimated XML Schema type T. When type information is available, we use ad hoc measures for computing similarity

between basic data types, as illustrated in Example 1. Class Builder supports a number of string distances like the Levenshtein (or edit) distance L, defined for two strings s and r of arbitrary length as the minimum number of character inserts, deletes, and changes needed to convert r into s.

The second function F aggregates the individual fx values, $F(f_a, f_b, f_c)$ and expresses the overall similarity between two XML documents. Using F as a comparison measure, we can perform content-based clustering using the same algorithm used for structural clustering. When choosing F, we need to take into consideration the tags' different semantic relevance. At one extreme, there is the situation in which all tag-level similarity values contribute to the overall similarity via a conjunction of their individual values. Here, F will belong to the t-norm class of operators.

At the other extreme lies the situation in which the overall result considers exactly one of the individual similarity values, operating a disjunction via a t-conorm. The aggregation depicted represents the simplest possible situation, when two documents have exactly the same tags (with no duplicates) and the same membership degrees (the same positions). In order to deal with the general case, we need to answer two additional questions not addressed by classical techniques of information clustering and retrieval what happens when a tag is present in one of the documents being compared and not in the other and 2) what are the "right" tag-pairs to be compared? When a tag appears in a document (with single or multiple cardinality) and not in the other, we simply consider tag level similarity to be zero.

VI. TEST CASES

In general a test case is a set of test data and test programs and their expected results. A test case in software engineering normally consists of a unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, output and it validates one or more system requirements and generates a pass or fail.

P. Test Case 1(It is shown in Fig 3)

Summary : Checking with unknown Site name
Initial Condition : User has to enter all the Information
Steps to Run : Run the program
Expected Output : A Message Dialog Window will be Displayed with dialog "Failed to catch" ,"Could not Receive link"
Actual Output : The Output is as Shown in the diagram

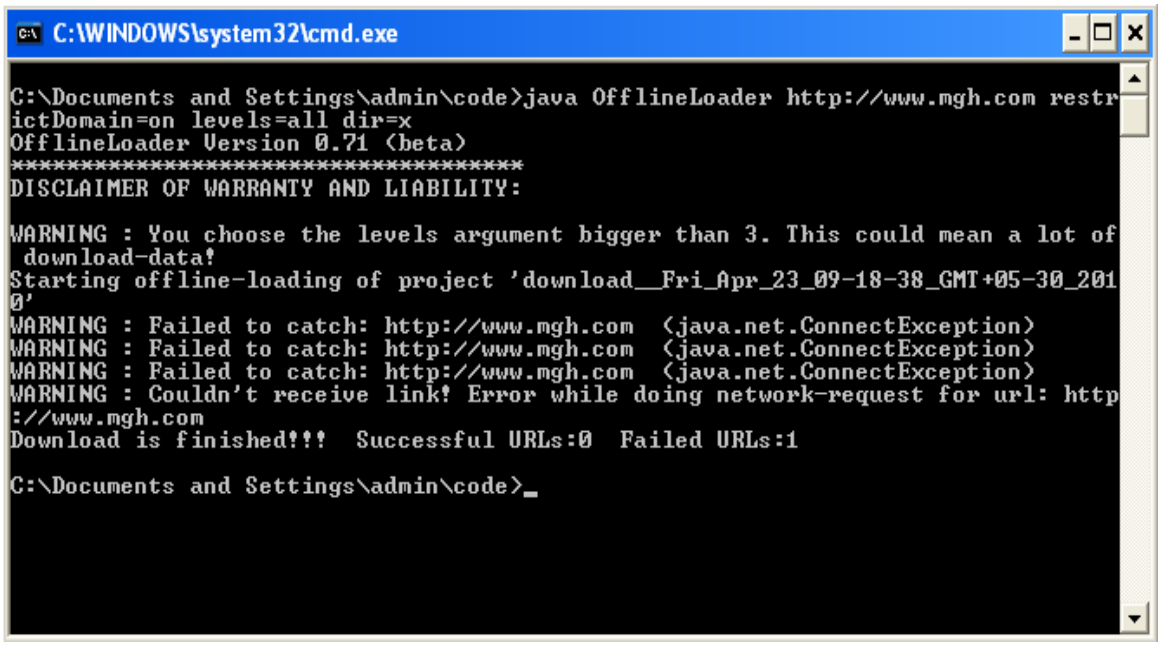

Fig 3 **Test case 1**

Q. Test Case 2(It is shown in Fig 4)

Summary : Checking with Unknown levels
Initial Condition : User has to Enter all the Information
Steps to Run : Run the program
Expected Output : A Message Dialog Window will be Displayed with dialog"You Choose the levels argument Bigger than 3"
Actual Output : The Output is as Shown in the diagram

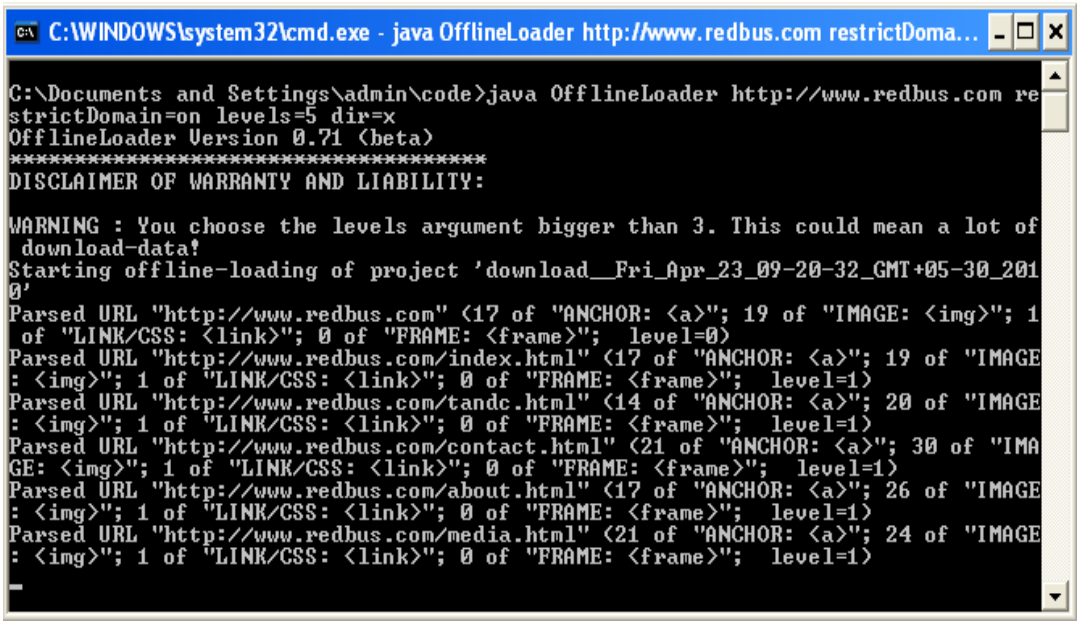

Fig 4 Test case 2

R. Test Case 3(It is shown in Fig 5)

Summary : Checking with restrict Domain is off
Initial Condition : User has to Enter all the Information
Steps to Run : Run the program
Expected Output : A Message Dialog Window will be Displayed with dialog"
"Levels is all domain is not restricted this means an infinity download"
Actual Output : The Output is as Shown in the diagram

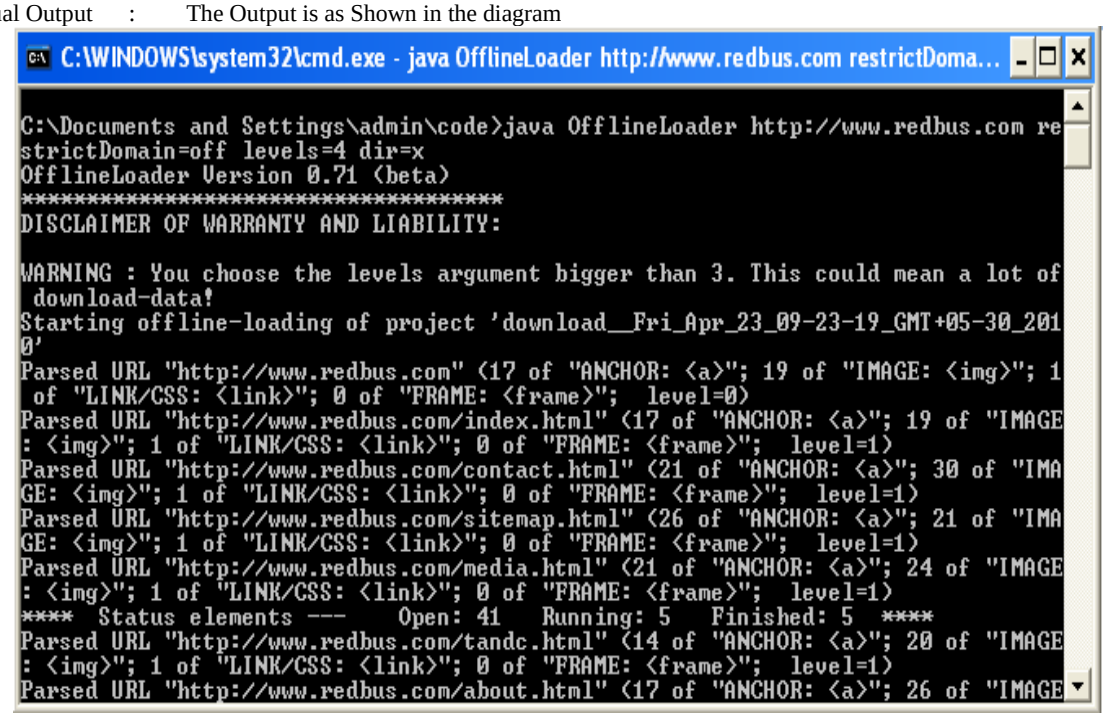

Fig 5 Test case 3

VII. CONCLUSIONS

The paper intended to interact with the user through the GUI, allowing him to easily deal with the web site as a whole. Although this application has not got some of the features present in the existing systems, it has incorporated some new features. This application helps the user in getting the broad picture of the web site and aimed at providing a set of services, which can help the user to deal with the remote web site in an easy and flexible manner. The application developed has been able to satisfy most of the requirements, initially drawn out during the problem description phase. The following conclusions can be drawn from the development of the project

- It provides an easy tool available for a browser to deal with sites while searching for information.

- It overcomes the delay caused in establishing connections if the web site is already downloaded.
- It can help in sharing of the web site over the intra net

Future Enhancements: To improve the functionality and usefulness of the application, the following enhancements can be made

- Enabling the application to handle any type of protocol. i.e. shttp, gopher, FTP etc.
- To let the user interpret any type of file that can be displayed in the internal browser.
- Automatic download of the files whose contents have changed based on the date of last modification
- Proving the user allowing him to use the proxy server.

REFERENCES

[1]. G. Burnett, M.H. Dickey, M.M. Kazmer, and K.M. Chudoba, "Inscription and Interpretation of Text: A Cultural Hermeneutic Examination of Virtual Community," Information Research, vol. 9, no. 1, 2003.

[2]. E. Lesser and K. Everest, "Using Communities of Practice to Manage Intellectual Capital," Ivey Business J., vol. 65, no. 4, pp. 37-41, 2001.

[3]. A. Farquhar, R. Fikes, and J. Rice, "The Ontolingua Server: A Tool for Collaborative Ontology Construction," Int'l J. Human-Computer Studies, vol. 46, no. 6, pp. 707-727, 1997.

[4]. G. Bordogna and G. Pasi, "A User-Adaptive Indexing Model of Structured Documents," Proc. 10th Int'l Conf. Fuzzy Systems, pp. 984-989, 2001.

[5]. E. Damiani, L. Tanca, and F. Arcelli Fontana, "Fuzzy XML Queries via Context-Based Choice of Aggregations," Kybernetika, vol. 36, no. 6, pp. 605-616, 2000.

[6]. V. Carchiolo, A. Longheu, and M. Malgeri, "Hidden Schema Extraction in Web Documents," Proc. Third Int'l Workshop Databases in Networked Information Systems (DNIS '03), pp. 42-52, 2003.

[7]. P. Ceravolo and E. Damiani, "Fuzzy Mining Class Hierarchies from XML-Based Authentication Data," Ontology Learning from Text: Methods, Applications, Evaluation, Sept. 2004.

[8]. M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New Algorithms for Fast Discovery of Association Rules," Proc. Third Int'l Conf. Knowledge Discovery and Data Mining (KDD '97), 1997.

[9]. P. Ceravolo, "Extracting Role Hierarchies from Authentication Data Flows," Int'l J. Computer Systems Science and Eng., vol. 4, no. 6, 2004.

[10]. I. Horrocks, U. Sattler, and S. Tobies, "Practical Reasoning for Expressive Description Logics," Proc. Sixth Int'l Conf. Logic Programming and Automated Reasoning (LPAR '99), pp. 161-180,1999.

[11]. K.M. Gupta, D.W. Aha, and P. Moore, "Learning Feature Taxonomies for Case Indexing Advances in Case-Based Reasoning," Proc. Seventh European Conf., 2004.

[12]. E. Damiani, M.G. Fugini, and C. Bellettini, "A Hierarchy-Aware Approach to Faceted Classification of Objected-Oriented Components," ACM Trans. Software Eng. Methodologies, vol. 8, no. 3, pp. 215-262, 1999.

[13]. M. Cristani and R. Cuel, "A Survey on Ontology Creation Methodologies," Int'l J. Semantic Web and Information Systems, vol. 1, no. 2, pp. 49-69, 2005

[14]. N. Fuhr and G. Weikum, "Classification and Intelligent Search on Information in XML," IEEE Data Eng. Bull., vol. 25, no. 1, pp. 51-58, 2002.

[15]. P. Kilpela¨inen, "Tree Matching Problems with Applications to Structured Text Databases," PhD dissertation, Dept. of Computer Science, Univ. of Helsinki, 1992.

[16]. B. Bouchon-Meunier, M. Rifqi, and S. Bothorel, "Towards General Measures of Comparison of Objects," Fuzzy Sets Systems, vol. 84, no. 2, pp. 143-153, 1996.

[17]. E. Damiani, M.C. Nocerino, and M. Viviani, "Knowledge Extraction from an XML Data Flow: Building a Taxonomy Based on Clustering Technique," Proc. EUROFUSE 2004: Eighth Meeting EURO Working Group on Fuzzy Sets, pp. 133-142, 2004.